Uppdatering och vidareutveckling av ett webbaserat intresseanmälningssystem

Lars Wikberg

Examensarbete för ingenjörs (YH)-examen

Utbildningsprogrammet för informationsteknik

Vasa 2014

EXAMENSARBETE

Författare: Lars Wikberg

Utbildningsprogram och ort: Informationsteknik, Vasa

Handledare: Mikael Jakas

Titel: Uppdatering och vidareutveckling av ett webbaserat intresseanmälningssystem

---

Datum: 12.05.2014 Sidantal: 23

Abstrakt

Detta arbete har utförts åt företaget Anvia i Vasa. Målet var att förnya och uppdatera en befintlig webbsida för kunder med intresse för fiberanslutning.

Arbetet utfördes med modern webbprogrammering efter diskussion och planering med uppgiftsgivaren Anvia.

Resultatet av arbetet är en ny webbsida där kunder kan söka på en gatuadress för att se om adressen är inom ett fiberområde. Om inte så kan kunden skicka in en intresseanmälan.

Språk: Svenska Nyckelord: intresse, PHP, JavaScript, MySQL

BACHELOR'S THESIS

Author:	Lars Wikberg
Degree Programme:	Information Technology, Vasa
Supervisors:	Mikael Jakas

Title: Updating and further development of a web-based interest notification system

---

Date: 12.05.2014 Number of pages: 23

Abstract

This thesis work was made for the company Anvia located in Vasa. The end goal was to renew and update an already existing web page for customers with interest in a fiberoptic internet connection.

The work was done with modern web programming after planning and discussion with Anvia.

The work resulted in a new web page where customers can search for a streetaddress to see if in a fiberoptic area. If not, the customer can send an interest notification.

Language: Swedish Key words: interest, PHP, JavaScript, MySQL

Innehållsförteckning

Ordförklaringar

API	Application Programming Interface är en regeluppsättning för hur en viss programvara kan kommunicera med en annan programvara.
UML	Unified Modeling Language är ett objektorienterat generellt språk för modellering av alla typer av system.
CSS	Cascading Style Sheets är ett stilmallsspråk som används för att definiera utseendet på ett dokument.
HTML	Hypertext Markup Language är ett märkspråk för hypertext
JSON	JavaScript Object Notation är ett kompakt, textbaserat format för datorer för att utbyta data.
jQuery	Ett JavaScript bibliotek tänkt för att förenkla webbutveckling.
SOAP	Ett protokoll för utbyte av information i decentraliserade och distribuerade miljöer.
SQL	Structured Query Language är ett standardiserat programspråk för att hämta och modifiera data i en relationsdatabas.
REST	Representational State Transfer är ett IT-arkitekturbegrepp som beskriver ett möjlig kommunikationssätt mellan maskiner.

1 Inledning

Arbetet gick ut på att planera och utveckla en webbsida som skulle samla in information om fiberintresserade. Intresserade skulle ha möjlighet att ta reda på om de hade möjlighet till fiberanslutning, om fiber höll på att byggas i närheten eller om det var planerat att bygga fiber i området. Detta skulle förverkligas genom att visa en karta på webbsidan. Informationen som samlas in av de fiberintresserade skulle också visas på webbsidan i form av anonym statistik.

1.1 Uppdragsgivare

Arbetet utfördes åt Anvia. Ab Wasa Telefonförening grundades 1882 som det första telefonbolaget i Vasa och har sedan dess ständigt växt genom företagsuppköp och utvidgning till andra teknikområden. Anvia består av Vasa Läns Telefon Ab (VLT) som bytte namn 2008 efter att man hade förnyat koncernens målsättning. Den nya målsättningen var att ständigt utvecklas och ha bra kontakt med kunderna. I dag består Anvia av tre affärsområden med ca 700 anställda och företagets totala omsättning var 108 miljoner euro år 2010. De tre affärsområden är Securi, ICT och TV. Anvias logo synns i figur 1.

Figur 1. Anvias logo.

1.2 Bakgrund

Anvia hade redan från tidigare i bruk en webbsida där man kunde anmäla sitt intresse om fiberanslutning till hemmet. Den gamla sidan skulle förnyas utseendemässigt, men behålla samma funktioner som den gamla för att fortsätta samla information om de intresserade på samma sätt som förut. Från informationen som samlats sedan den gamla sidan togs i bruk skulle anonym statistik framtas för att visas på webbsidan. De nya intresseanmälningarna

skulle insättas till samma databas som redan var i bruk, så att de automatiskt blev inkluderade i statistiken på webbsidan. Förutom att anmäla intresse om fiberanslutning, skulle kunden kunna lämna anbudsbegäran om kunden befann sig inom ett av Anvias fiberområden.

En karta skulle tillsättas för att visa geografiskt var adresserna fanns som intresserade sökte på och visa om där fanns fiber tillgängligt, om fiber är planerat i området eller om fiber inte är tillgängligt.

2 Tekniker

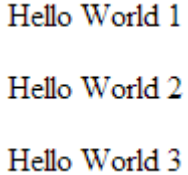
Här listas de tekniker som användes för att uppnå målsättningen. På klienten användes HTML, CSS och JavaScript. På servern användes PHP och MySQL.

2.1 HTML

Hypertext Markup Language. HTML är ett märkspråk för hypertext som är en av de grundläggande standarderna för WWW. Med protokollen HTTP och TCP/IP överförs HTML-baserade sidor från servrar till webbläsare. HTML är utmärkt för att skapa statiska webbsidor. Figur 2 visar HTML-koden från kodexempel 1. (Beighley & Morrison, 2009, s. 2)

Kodexempel 1. HTML

```
<!DOCTYPE html>
<html>
	<head>
	</head>
	<body>
		<p>Hello World 1</p>
		<p>Hello World 2</p>
		<p>Hello World 3</p>
	</body>
</html>
```



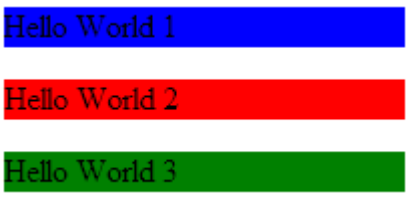
Figur 2. Renderad HTML-kod från kodexempel 1

2.2 CSS

Cascading Style Sheets, CSS är ett språk som beskriver presentationsstilen för ett strukturerat dokument. Utvecklat för att kunna separera själva innehållet av strukturerade dokument från presentationen av dokumentet. Layouten, färger och fonter kan hållas separat från HTML. CSS kan också deklareras inne i HTML, som syns i kodexempel 2. Den slutliga sidan från kodexempel 2 visas i figur 3. (About jQuery, 2013)

Kodexempel 2. CSS

```
<!DOCTYPE html>
<html>
	<head>
		<style type="text/css">
			p { background-color: blue; }
			p.two { background-color: red; }
			p.three { background-color: green; }
		</style>
	</head>
	<body>
		<p>Hello World 1</p>
		<p class="two">Hello World 2</p>
		<p class="three">Hello World 3</p>
	</body>
</html>
```



Figur 3. Koden från kodexempel 2 som visad av en webbläsare.

2.3 Javascript

Javascript är ett skriptspråk som främst används i webbläsare för att på klientsidan manipulera innehållet. JavaScript skrivs i Unicode teckenuppsättningen, som är en överordnad av ASCII och Latin-1 och stöder virtuellt varje skrivet språk som används på planeten. JavaScript är versalkänsligt. Eftersom HTML inte är versalkänsligt kan detta vara förvirrande. Kodexempel 3 visar hur man kan skriva till ett HTML-dokument. Figur 4 visar resultatet från kodexempel 3 när det renderats av en webbläsare. (Flanagan, 2011, s. 21)

Kodexempel 3. JavaScript

```
<!DOCTYPE html>
<html>
	<head>
		<style type="text/css">
			p { background-color: blue; }
			p.two { background-color: red; }
			p.three { background-color: green; }
		</style>
	</head>
	<body>
		<p>Hello World 1</p>
		<p class="two">Hello World 2</p>
		<p class="three">Hello World 3</p>
		<script type="text/javascript">
			document.write( '<b>Hello JavaScript World<b>' );
		</script>
	</body>
</html>
```

Figur 4. Koden från kodexempel 3 visad i en webbläsare.

2.3.1 JQuery

Jquery är ett öppet javascript-bibliotek under MIT- och GNU-licens utvecklat för att förenkla skriptandet på klientsidan av webbsidor. jQuery är det populäraste biblioteket idag och används på över hälften av de 10 000 mest besökta webbsidorna. Designat för att enklare navigera på en sida, enkelt skapa animationer, hantera event och behandla DOM element. jQuery erbjuder också goda möjligheter att utveckla insticksprogram till biblioteket, vilket har gjort det populärt bland programmerare. I kodexempel 4 används jQuery för att ändra bakgrundsfärgen på alla HTML b element till gul. Resultatet syns i figur 5. (About jQuery, 2013)

Kodexempel 4. jQuery

```
<!DOCTYPE html>
<html>
	<head>
		<script src="http://code.jquery.com/jquery-
latest.js"></script>
		<style type="text/css">
			p { background-color: blue; }
			p.two { background-color: red; }
			p.three { background-color: green; }
		</style>
	</head>
	<body>
		<p>Hello World 1</p>
		<p class="two">Hello World 2</p>
		<p class="three">Hello World 3</p>
		<script type="text/javascript">
			document.write( '<b>Hello JavaScript World<b>' );
		</script>
		<script type="text/javascript">
			$( 'b' ).css( 'background-color', 'yellow' );
		</script>
	</body>
</html>
```

Figur 5. Renderade sidan från kodexempel 4.

2.3.2 AJAX

Asynchronous JavaScript and XML. Med AJAX kan man skicka och ta emot data från en server i bakgrunden utan att behöva ladda om sidan. År 2003 tillsattes de stora webbläsaraktörerna XMLHttpRequest (XHR) som möjliggjorde att webbläsare kunde kommunicera med webbservern utan att behöva ladda om webbsidan. XHR är en del av AJAX. AJAX fungerar med JavaScript kod som sänder en förfrågning till en URL. När den får svar kallas en callback funktion för att hantera datan i svaret. (About jQuery, 2013)

2.3.3 JSON

JavaScript Object Notation är ett textformat för att utbyta data. Skapades efter JavaScripts sätt att representera datastrukturer och arrayn som ett mänskligt läsbart sätt att utbyta data. Används ofta som ett alternativ till XML genom att erbjuda samma egenskaper vid dataöverföring över nätverk. Används oftast mellan en server och en webapplikation. (About jQuery, 2013)

2.4 PHP

PHP är ett skriptspråk som främst används på webbservrar med dynamiskt innehåll. Det som idag är PHP hade sin början 1995 när Rasmus Lerdorf utvecklade ett Perl/CGI skript som loggade sidvisningar och visade antalet sidvisningar på hans egna hemsida. Efter många förfrågningar släppte Lerdorf sitt skript som hand döpte till Personal Home Page, PHP. Efter skriptens goda mottagande valde Lerdorf att vidareutveckla PHP och började med att byta till C. I figur 6 syns resultatet från kodexempel 5. (Beighley & Morrison, 2009, s. 2)

Kodexempel 5. PHP

```
<?php
      echo 'Hello World.';
?>
```

Hello World.

Figur 6. Utskriften från kodexempel 5.

2.5 MySQL

My Structured Query Language, MySQL, är en fri databashanterare under GNU-licens. Den använder sig av frågespråket SQL. MySQL är ett populärt val av databas för webbaserade lösningar och är en del av programvarustacken LAMP. LAMP är ett akronym till Linux, Apache, MySQL och PHP/Perl/Python. Se kodexempel 6 och figur 7 för ett exempel på MySQL-förfrågan och tillhörande resultat.

Kodexempel 6. MySQL

```
SELECT 'Hello world';
```

```
+-------------+
| Hello World |
+-------------+
| Hello World |
+-------------+
1 row in set (0.00 sec)
```

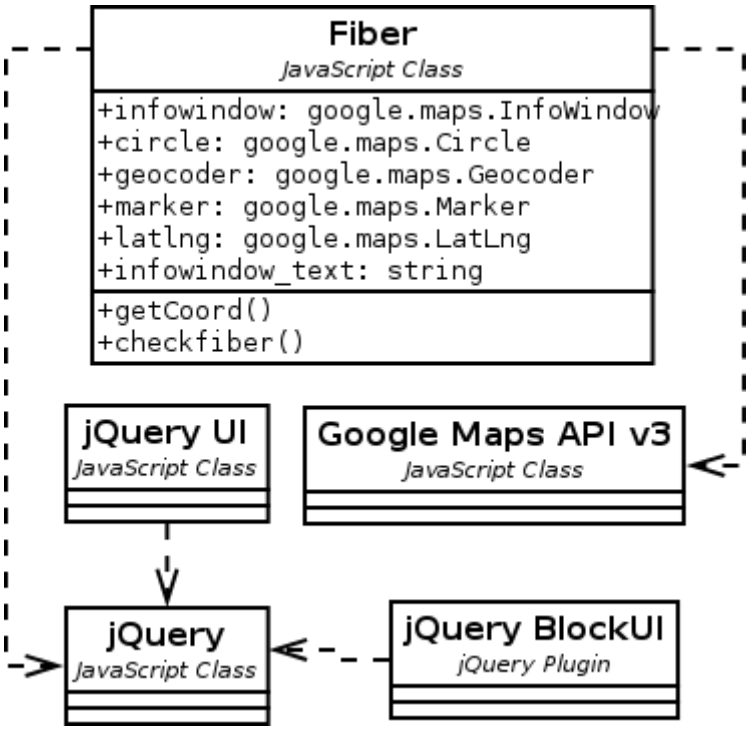
Figur 7. MySQL resultatet från kodexempel 6.

3 Utförande

Utförandet av projektet gick till på följande sätt. Den första delen behandlar logiken och funktionaliteten på klientsidan. Den andra delen behandlar det som körs på serversidan av webbsidan och i den tredje delen berättar om testningen och användningen av olika verktyg medan sidan utvecklades.

3.1 På klienten

Det mesta på webbsidan i JavaScript hanteras i en huvudklass Fiber, vars UML-klassdiagram kan ses i figur 8. Fiberklassen är beroende av jQuery, jQueryUI och Google Maps API. Förutom det används också en jQuery plugin BlockUI.



Figur 8. UML-klassdiagram över JavaScript-klassen Fiber.

3.1.1 Webbsidan

Figur 9 är en översikt av webbsidan när man först hämtat den. Överst till höger finns en länk för att byta till det andra inhemska språket. Mitt på sidan kan man söka på sin gatuadress, vill man precisera sökningen mera kan man också fylla i postnummer. Under sökrutan visas en karta och på höger sida visas tre tabeller med statistik från tidigare fiberintresserade. Längst ner på sidan finns länkar till Anvias sidor med mera information om fiber, kontaktuppgifter till Anvia och delningsmöjligheter till flera olika sociala medier.

Figur 9. Översikt av webbsidan vid start.

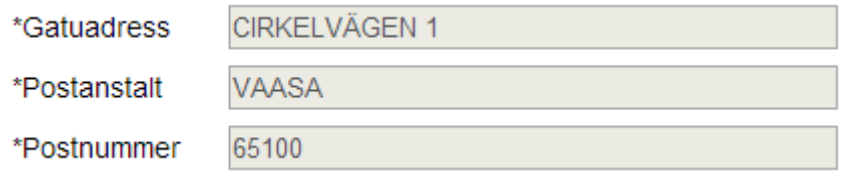
Besökaren får söka på önskad adress. Gatuadressfältet använder jQuerys funktion autocomplete för att autokomplettera matchande adresser. Som källa till autokompletteringen används Anvias egna nätverksregister. Detta genom att sätta autocomplete-funktionens källa till en PHP-fil på servern (Adressautokomplettering) och skicka med adressträngen och postnumret med hjälp av AJAX. PHP-filen tar emot adressträngen som skickades och de matchande adresserna returneras i JSON. Postkoden skickas endast med om den består av fem tecken. Med hjälp av det returnerade JSON-dokumentet kan autocomplete funktionen visa en lista, se figur 10, för användaren på matchande adresser.

Kodexempel 7. Källan sätts i jQuerys autocomplete funktion.

```
$( "#addressautocomplete" ).autocomplete( {
      source: function( request, response ) {
            var zippy = ( $( '#addressautocomplete_zip' ).val().length ==
5 ) ? $( '#addressautocomplete_zip' ).val() : "";
            $.ajax( {
                  url: "getaddressautocomplete.php",
                  dataType: "json",
                  data: {
                        term: request.term,
                        zip: zippy
                  },
                  success: {…}
                  …
```

Figur 10. Exempel på autokompletteringsresultat.

När besökaren väljer en adress i autocomplete-listan fylls gatuadressen, vägnumret, orten och postnumret i färdigt i formuläret och fälten inaktiverade, se kodexempel 8 och figur 11. Fastän orten har både svenskt och finskt namn används alltid det finska, medan gatuadressen kan ha både svenskt och finskt namn. Detta beroende på om den sökande angav det svenska eller finska vägnamnet vid sökningen.



Figur 11. Adressuppgifterna färdigt ifyllda och inaktiverade i formuläret.

Kodexempel 8. Kod som körs när en adress väljs i jQuerys autocomplete-funktion.

```
select: function( event, ui ) {
	// Adressen fylls i färdigt i formen efter val.
	$( '#postiosoite' ).val( ui.item.street );
	$( '#postitoimipaikka').val( ui.item.postoffice );
	$( '#postinumero').val( ui.item.zip );
	// Dolda form detaljer sätts efter val.
	$( '#streetaddress' ).val( ui.item.street );
	$( '#postcode').val( ui.item.zip );
	$( '#zip').val( ui.item.zip );
	$( '#postoffice').val( ui.item.postoffice );
	$( '#buildingid' ).val( ui.item.buildingid );
	$( '#communitycode' ).val( ui.item.districtcode );
	// Koordinaterna till adressen hämtas.
	fi.getCoord( function( ll ) {
		fi.latlng = ll;
		// Söker om fiber finns på koordinaterna.
		fi.checkfiber();
	} );
}
```

Sedan kallas funktionen getCoord, se kodexempel 9, som med hjälp av regex endast tar med gatunamnet och vägnummern i sökningen, eftersom Google Geocode inte har så specifik information som byggnaders trappuppgångar eller dörrnummer. Till gatunamnet och vägnummret sätts också postnumret och Finlands förkortning FI för att begränsa sökningen endast till Finland.

Kodexempel 9.

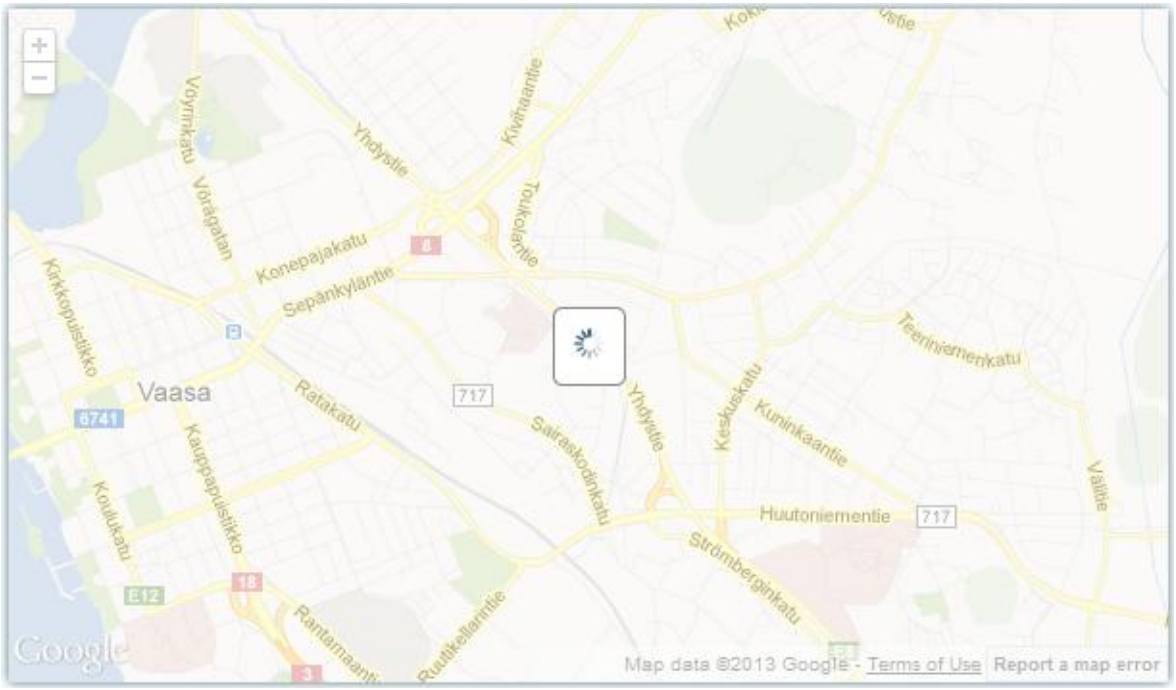
```
this.getCoord = function( callbacky ) {
	var street = $( '#streetaddress' ).val();
	var regex = /([A-å]*) ([0-9]*-*[0-9]*[A-z]?) (.*)/i;
	// Tar bort allt efter vägnamn och vägnummer.
	street = street.replace( regex, "$1 $2" );
	var zip = $( '#zip' ).val();
	// Lägger till postnummer och FI till adressen
	var address = street + ", " + zip + ', FI';
	this.geocoder.geocode( { 'address': address }, function( results, status )
{
		if( status == google.maps.GeocoderStatus.OK ) {
			callbacky( results[0].geometry.location );
		}
		else if( status == google.maps.GeocoderStatus.ZERO_RESULTS ) {
			alert( address + ' was not found' );
		}
		else {
			alert( 'Geocode was not successful for the following reason:
' + status );
		}
	} );
}
```

När Google Geocode returnerar koordinaterna till adressen så sätts koordinaterna till kartans centrum och kartan zoomas in.

En PHP-fil (points.php) hämtas från webbservern med jQuerys getJSON-funktion med koordinaterna som parametar. Medan points-filen hämtas och laddas används jQuerys plugin BlockUI för att blockera kartan på sidan och visar en låda i mitten ovanpå kartan med en animerad ikon för att indikera att sökning pågår.

Kodexempel 10. jQuery BlockUI

```
$( '#map' ).block( {
      message: $( '#loadingdiv' ).html(),
      applyPlatformOpacityRules: false,
      overlayCSS: {
            backgroundColor: '#fff'
      },
      css: {
            border: "1px solid gray",
            width: "auto",
            padding: "10px",
            '-moz-border-radius': "5px",
            '-webkit-border-radius': "5px",
            'border-radius': "5px"
      }
} );
```



Figur 12. BlockUI med en animerad ikon indikerar att information hämtas i bakgrunden.

I kodexempel 11 hämtas PHP-filen (points.php) med bifogade koordinater. PHP-filen returnerar ett JSON-dokument med ett JavaScript-objekt med egenskaper som berättar om fiber är tillgängligt vid de angivna koordinaterna, se figur 13. Jquery-pluginen BlockUI avaktiveras över kartan, se figur 12, när PHP-filen laddats klart. Egenskapen fiber sätts till en etta om det finns fiber, annars en nolla. Color-egenskapen returnerar en HTML-färg och bestämmer vilken färg cirkeln som ritas på kartan ska få.

Kodexempel 11. jQuerys funktion getJSON hämtar en fil.

```
$.getJSON( 'points.php?q=' + this.latlng, function( data, textstatus ){…}
```

```
color: "#28EF0E"
fiber: 1
```

Figur 13. Exempel på returnerade JSON-dokument från points.php.

Ett Google Maps InfoWindow används för att informera om det finns fiber tillgängligt på adressen eller inte, se figur 14. Formuläret som varit gömt med CSS-regler sedan sidan laddades, visas nu med jQueryUIs funktion slideDown, som animerar en nedrullning av formuläret under kartan. I kartans informationsfönster finns också en länk till formuläret.

Figur 14. En cirkel och ett informationsfönster berättar om fibertillgängligheten.

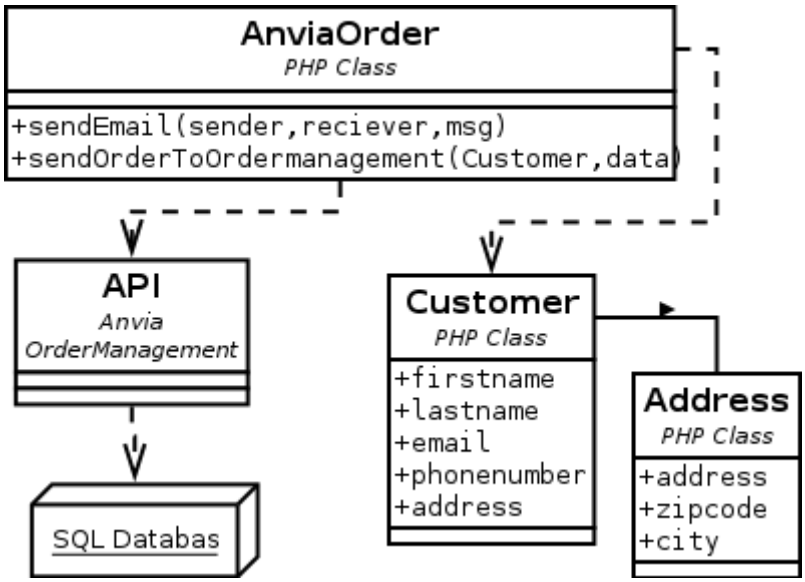
Formuläret ska skickas till olika interna tjänster beroende på om fiber är tillgängligt på adressen eller inte. Formuläret är det samma men med olika rubriker för de båda fallen, så att rätt avdelningar inom Anvia får informationen på rätt ställe. Om det inte går att få fiber på adressen så ska formuläret insättas i databasen för fiberintresserade. Om fiber finns och kunden väljer att skicka anbudsbegäran, skickas formuläret direkt till Anvias orderhanteringssystem.

3.2 På servern

På serversidan används i huvudsak tre PHP-klasser. Dessa tre klasser sköter om kommunikationen med tre av Anvias interna APIn. Förutom dessa tre använde jag också en av Anvia tidigare utvecklad PHP-klass som sköter om översättningen på sidan.

3.2.1 Anvias orderhanteringssystem

AnviaOrder PHP-klassen används att skicka formuläret till Anvias interna orderhanteringssystem och för att skicka konfirmation till kunden via e-post efter att kunden skickat formuläret. Konfirmationen att Anvia har tagit emot formuläret skickas också fast formuläret inte skickades till orderhanteringssystemet, utan också när det skickas till Anvias fiberintressetabell. I figur 15 visas ett UML-diagram över AnviaOrder-klassen.

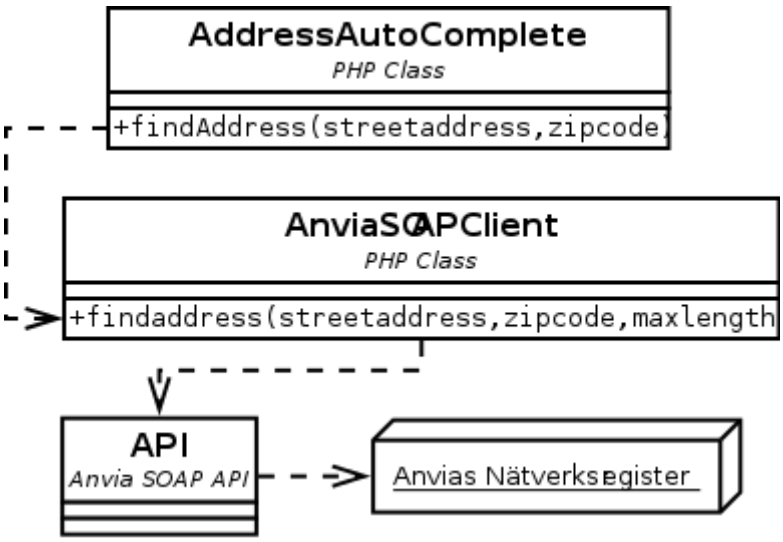


Figur 15. UML-diagram över orderhanteringen.

3.2.2 Adressautokomplettering (getadressautocomplete.php)

En PHP-klass som sköter om autokompletteringen till adressfältet. Klassens UML-diagram kan ses i figur 16. Tar emot en gatuadress och alternativt en postkod som parametrar för att söka genom Anvias interna nätverksregister. Returnerar matchande adresser.

Skickar gatuadressen och om angetts postkoden till Anvias SOAP API som returnerar matchande adresser ur Anvias nätverksregister. Med Anvias SOAP-klient fås svaret från SOAP API:n som en PHP-array och kan konverteras till JSON med PHP-funktionen json_encode för att sedan skrivas ut med echo.

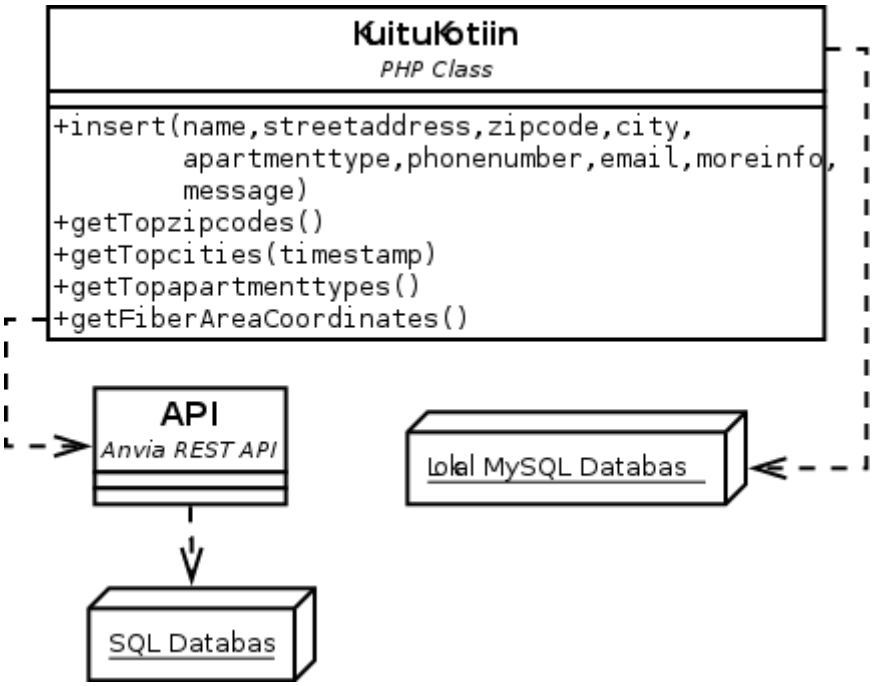


Figur 16. UML-diagram över adressautokompletteringen.

3.2.3 Fiberintresse (points.php)

PHP-klassen fiberintresse sköter om kommunikationen med den lokala SQL-databasen och kommunikationen med Anvias REST API. Ett UML-diagram över klassen finns i figur 17. Klassen är i PHP-filen points.php. Points.php tar emot koordinaterna med PHP:s fördefinierade variabel _GET. Points.php returnerar JSON med egenskaper om det finns fiber eller inte och om det är inom ett fiberområde en färgkod beroende på vilken typ av fiberområde det är frågan om.

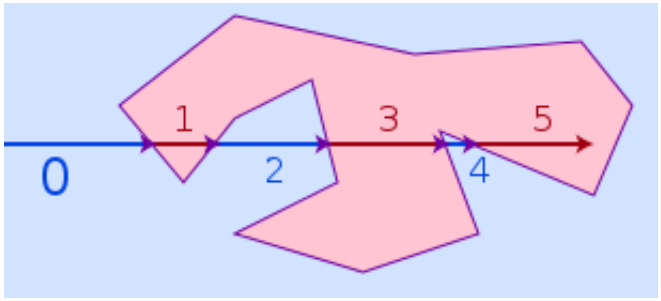
Från Anvias REST API fås alla kända fiberareor, som består av tvådimensionella polygoner av koordinater.



Figur 17. UML-diagram.

3.2.3.1 Punkt i polygon (Ray Casting Algorithm)

Ett enkelt sätt att bestämma om en punkt finns inom en polygon är att testa hur många gånger en linje, med start från punkten, i valfri riktning korsar polygonens gränser, visualiserat i figur 18. Om punkten är inom polygonen passerar den polygonens gränser ett ojämnt antal gånger och om den är utanför polygonen passerar den jämnt antal gånger.



Figur 18. Polygon.

Algoritmen är så pass vanlig inom beräkningsgeometri, att för de vanligaste programmeringsspråken finns det massvis med färdiga funktioner på Internet som berättar om en punkt är inom en polygon, se kodexempel 12.

Kodexempel 12. PHP-kod för kontroll om en punkt är inom en viss area.

```
if( pointInPolygon($point,$area)== "inside" ) {}
```

3.2.4 Anvia SOAP API

Anvias interna SOAP API används för att hämta adresser ur Anvias nätverksregister till autokompletteringen på webbsidan. Nätverksregistret fanns på en annan server. Servern hade en SOAP-server som man kunde använda för att skicka förfrågningar till nätverksregistret. SOAP-server hade redan en funktion för att söka efter adresser och för att kommunicera med Anvias nätverksregister användes en av Anvias tidigare utvecklade extension av PHP-klassen SoapClient.

Kodexempel 13. Initierar SOAP-klienten och skickar en förfrågning.

```
require_once( 'AnviaSoapClient.class.php' );
$client = new AnviaSoapClient( 'anvia.tigers.findaddress' );
try {
	$addresses = $client->findAddress( $searchterm, $zipcode, 200 );
}
catch( Exception $e ){
	return;
}
```

3.2.5 Anvia REST API

Anvias interna REST API används för att hämta koordinaterna till alla Anvias fiberområden. För att kommunicera med REST-servern användes PHP-klassen HTTP_Request2 för att posta vissa fördefinierade parametrar till servern, så servern vet att det är ett internt verktyg från Anvia som skickar förfrågan. Man anger också vilken metod det är man vill ska köras. I det här fallet vill vi ha alla fiberareor och då ser förfrågan ut som i kodexempel 14.

Kodexempel 14. Förfrågning till Anvias REST API.

```
$request = new HTTP_Request2();
$request->setMethod( HTTP_Request2::METHOD_POST );
$request->addPostParameter( 'apikey', 'secretanviakey' );
$request->addPostParameter( 'method', 'getfiberareacoordinates' );
$response = $request->send();
```

3.2.6 Databas och statistik

Anvias förra sida hade samlat namn med tillhörande adresser på de som anmält sig som fiberintresserade redan i två års tid. Strukturen för databastabellen som används syns i figur 19. Med hjälp av den redan insamlade datan skulle några tabeller skapas med anonym statistik. Vilken statistik som skulle visas fick jag bestämma. Tre olika tabeller valdes. Postnummerområden där intresset är störst, intresse enligt bostadstyp och orter där intresset är störst.

Webbsidan som Anvia använde för att samla fiberintresserade förut hade ingen granskning för att se om de intresserade fyllde i korrekta adressuppgifter. Orter och vägnamn med både finska och svenska namn eller en gatuadress i postanstaltfältet försvårade visningen av statistiken över hela perioden.

Eftersom orter fanns i databasen på både finska och svenska valdes att för hand byta namn på de svenska kommunera till finska. Men eftersom här bara visas de fem orter med flest intresserade byttes namn på alla, utan bara på de populäraste. Detta problem försvinner med den nya webbsidan, eftersom man kan endast välja adresser som finns i Anvias egna register och den nya sidan använder också alltid det finska namnet på orter.

Figur 19. Tabellstrukturen för fiberintresserade.

3.2.6.1 Postkodsstatistik

Postkodsstatistik för att visa vilka postkoder som har flest intresserade från alla olika postnummer över hela tiden data samlats in. Procentuell visning av de fem postnummer med flest intresserade. För att hämta denna statistik ur databasen använde jag en SQL-förfrågning med en nästlad förfrågning. Hela SQL-förfrågningen kan ses i kodexempel 15 och resultatet i figur 20.

Kodexempel 15. SQL-förfrågning för postkoder med flest fiberintresserade procentuellt.

```
SELECT
      zipcode,
      LOWER( city ) AS city,
      ROUND(
            ( 100 * COUNT(*) ) / ( SELECT COUNT(*) FROM FiberInterest )
            , 1 ) AS count
FROM FiberInterest
GROUP BY zipcode
ORDER BY count DESC
LIMIT 5
```

60200	Seinäjoki	4.6 %
65280	Vasa	3.5 %
60320	Seinäjoki	3.2 %
66400	Laihela	3.0 %
65100	Vasa	2.9 %

Figur 20. Postkodsstatistik.

3.2.6.2 Statistik över bostadstyper

Anvia samlar in bostadstyp av de som anmält sitt fiberintresse. En procentuell fördelning av bostadstyperna från alla fiberintresserade visas. SQL-förfrågningen kan ses i kodexempel 16 och resultatet i figur 21.

Kodexempel 16. SQL förfrågning för procentuell fördelning av samtliga bostadstyper.

```
SELECT
      apartmenttype,
      ROUND( ( 100 * COUNT( apartmenttype ) ) /
            ( SELECT COUNT( * ) FROM FiberInterest )   , 1
      ) AS COUNT
FROM FiberInterest
GROUP BY apartmenttype
ORDER BY count DESC
```

Egnahemshus	75.7 %
Radhus	11.1 %
Höghus	9.8 %
Parhus	2.3 %
Affärslokal	1.0 %

Figur 21. Intresse enligt bostadstyp.

3.2.6.3 Statistik över orter

Flest intresserade enligt orter de senaste sex månaderna. Endast de fem orter med flest intresserade visas. SQL-förfrågningen kan ses i kodexempel 17 och resultatet i figur 22.

Kodexempel 17. SQL-förfrågning för orter med flest fiberintresse de senaste sex månaderna.

```
SELECT
      tstamp,
      LOWER( city ) AS city,
      ROUND(
            ( 100 * COUNT( city ) ) / ( SELECT COUNT(*) FROM
FiberInterest WHERE tstamp > DATE_SUB( NOW(), INTERVAL 6 MONTH )
            , 1 ) AS count
FROM FiberInterest
WHERE tstamp > DATE_SUB( NOW(), INTERVAL 6 MONTH )
GROUP BY city
ORDER BY count desc
LIMIT 5
```

Vasa	19.6 %
Seinäjoki	13.2 %
Karleby	7.9 %
Laihela	4.4 %
Hyllykallio	4.4 %

Figur 22. Statistik enligt orter.

3.2.7 Översättning

Översättning av webbsidan mellan finska och svenska skedde på servern med hjälp av en tidigare utvecklad PHP-klass från Anvia. Klassen bestod i huvudsak av en funktion som tog in en textsträng som parameter och beroende på vilket språk besökaren av webbsidan har valt, returnerar funktionen den motsvarande textsträngen på finska eller svenska. För

textsträngen som anges bör det också finnas en mapp med språkets förkortning med en fil som innehåller textsträngen. Där skriver man vad de enskilda textsträngarna ska returnera för de olika språken. Om översättningsfilerna från figur 23 i kodexempel 18 användes och användaren valt svenska som språk skulle funktionen returnera ”Optofiber intresserar”.

Kodexempel 18. Översättningsfunktionen.

```
<h1><?php echo msg( 'Kuitu kiinnostaa' ); ?></h1>
```

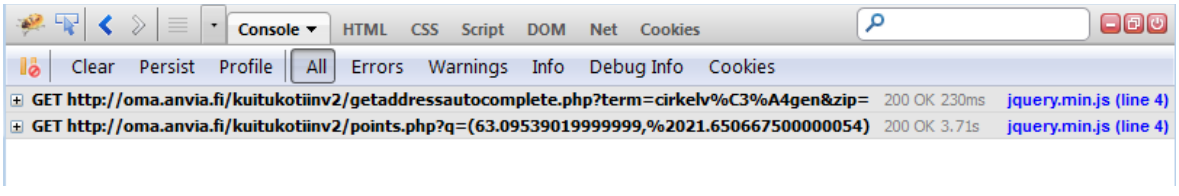
```
Kuitu kiinnostaa|Optofiber intresserar
Kuitu kiinnostaa|Kuitu kiinnostaa
```

Figur 23. Exempelrad ur översättningsfiler på svenska och finska.

3.3 Testning och verktyg

Testningen under utvecklingen av webbsidan skedde med de tre stora webbläsarna Google Chrome, Mozilla Firefox och Internet Explorer. Testningen med Chrome och Firefox var med de senaste versionerna, medan det bara fanns tillgång till Internet Explorer version 8 på den dator som användes under utvecklingen.

Som huvudprogram vid testning använde jag Firefox med extensionen Firebug som hjälper mycket vid utveckling av webbsidor. Man ser enkelt vilka filer som hämtas i bakgrunden och vad de har returnerat. Detta hjälpte mycket när Ajax användes mycket för att hämta filer i bakgrunden vid flera tillfällen. I figur 24 ser man firebug som visar vad som hämtats i bakgrunden av JavaScript efter att man först sökt på adressen Cirkelvägen och sedan valt Cirkelvägen 6 från listan.



Figur 24. Firebug visar vad som hämtats i bakgrunden.

4 Resultat

Resultatet av uppdraget är en förnyad webbsida för fiberintresserade. Webbsidan fick modernare layout med dynamiskt innehåll. De intresserade får genast se om de redan har tillgång till fiber genom adressökningen och den tillagda kartan. Anonym statistik för besökare av webbsidan samlas och visas för senare besökare.

4.1 Vidareutveckling

Redan vid planeringsskedet hade vi vissa punkter som skulle kunna utvecklas ännu efter att webbsidan lanserats i detta stadiet. Så som möjlighet att byta ut de nuvarande statistiktabellerna om bättre idéer på statistik dyker uppges. Att visa en tabell över de områden som det byggs fiber på för tillfället och en som visar områden som senast fått fiber. Ta i beaktande de fiberintresserade ur databasen som inte hade tillgänglighet till fiber när de visade intresse och fått tillgång till fiber i efterhand.
Kartdelen kunde utvecklas för att t.ex. säga på ett ungefär hur långt en adress är från det närmaste fiberområdet och i vilken riktning det ligger.

5 Diskussion

Det har varit roligt och lärorikt att arbeta med detta projekt, men utmanande. Detta då så mycket händer samtidigt i bakgrunden av webbsidan. Det tog länge innan jag själv fick en bra helhetsbild över allt som händer på webbsidan. Mina kunskaper inom dynamisk webbutveckling med hjälp av JavaScript och förbättrats mycket genom detta arbete. Kunskaper om hur olika API fungerar och hur man använder dessa med både JavaScript och PHP.

Källförteckning

Beighley Lynn & Morrison Michael. (2009) *Head First PHP & MySQL*. Sebastopol: O'Reilly Media

Flanagan David. (2011) *Javascript: The Definitive Guide, Sixth Edition.* Sebastopol: O'Reilly Media

Powers David. (2006) *PHP Solutions: Dynamic Web Design Made Easy.* New York: Springer-Verlag

About jQuery (2013)
http://learn.jquery.com/about-jquery/ (Hämtat: 9.4.2013)